AES

2020年3月26日 11:38

1. 群：是一个代数系统，它由一个非空集合G组成，在集合G上定义了一个二元运算，满足以下性质，则记$<G,\cdot>$为群：
    a. 封闭性：对任意的$a,b\in G$, $a\cdot b\in G$.
    b. 结合律：对任何的$a,b,c\in G$, $a\cdot b\cdot c=(a\cdot b)\cdot c=a\cdot(b\cdot c)$.
    c. 单位元：即存在一个元素$1\in G$(称为单位元)，对任意$a\in G$有$a\cdot 1=1\cdot a=a$。
    d. 逆元：对任意$a\in G$，存在一个元素$a^{-1}\in G$，使得$a\cdot a^{-1}=a^{-1}\cdot a=1$.
2. 特殊的群：
    a. 交换群：满足交换律的群。
    b. 有限群：包含有限多个元素，元素的个数成为阶。
3. 群的性质：
    a. 群中的单位元是唯一的；
    b. 消去律成立，即对任意的$a,b,c\in G$, $ab=ac$，则b=c;
    c. 群中的每一元素的逆元是唯一的。
4. 域：一个代数系统，有一个至少包含两个元素的非空集合F组成，在集合F上定义有两个二元运算：加法（用符号+ 表示）和乘法（用符号·表示），并满足下面条件，记为$<F,+,\cdot>$为域：
    a. F的元素关于加法‘+’成交换群，记其单位元为0（称为域的零元）；
    b. F关于乘法‘·’成交换群，记其单位元为1（称为域的单位元）；
    c. 乘法在加法上满足分配律，即对任意的$a,b,c\in F$，有$a\cdot(b+c)=ab+ac$，$(a+b)\cdot c=ac+bc$
5. 特殊的域：
    a. 若集合F只包含有限个元素，则称这个域F为有限域，也称为Galois域；有限域中的元素个数也称为该有限域的阶。
    b. 若有一任意的素数P和正整数$n\in Z^+$，存在P^n阶有限域，这个有限域记为GF(P^n)，当n=1时，有限域GF(P)称为**素域**。
    c. 域$<F,+,\cdot>$上x的多项式$a(x)=a_n x^n+a_{n-1} x^{n-1}+\cdots+a_1 x+a_0$，定义运算加法⊕和乘法⊗如下：
    d. 加法⊕：$a(x)\oplus b(x)=\sum_{i=0}^n a_i x^i \oplus\sum_{i=0}^m b_i x^i$
    $= \sum_{i=0}^M (a_i+b_i) x^i$;
    e. 乘法⊗：$a(x)\otimes b(x)=\sum_{i=0}^n a_i x^i \otimes\sum_{i=0}^m b_i x^i$
    $= \sum_{i=0}^{n+m} \left(\left(\sum_{j=0}^i a_j\cdot b_{i-j}\right) x^i\right)$;
    f. 其中$M=\max(m,n)$。
    g. 任何有限域都可以用与它同阶的多项式域表示。密码学中一般就是素域或阶为2^n的有限域。
6. $\boldsymbol{GF(2^8)}$域上的多项式表示及运算
    a. 表示：在AES加密系统中，$GF(2^8)$是不可约多项式$\boldsymbol{m(x)=x^8+x^4+x^3+x+1}$上构造的有限域$<F(x)_{m(x)},+,\cdot>$。

   i. 一个字节的$GF(2^8)$元素的二进制展开成的多项式系数为$b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$，即$b_7 x^7+b_6 x^6+b_5 x^5+b_4 x^4+b_3 x^3+b_2 x^2+b_1 x^1+b_0$
   ii. 例如: $GF(2^8)$上的37（十六进制），其二进制为00110111，对应多项式为$x^5+x^4+x^2+x+1$。
b. 加法：十六进制37+83=B4，采用二进制表示为00110111+10000011=10110100；
   i. 采用多项式为$(x^5+x^4+x^2+x+1)+(x^7+x+1)=x^7+x^5+x^4+x^2$
c. 模运算：37 mod 07 = 01；
   i. $(x^5+x^4+x^2+x+1) \bmod (x^2+x+1) = 1$
d. 乘法运算：57×83=C1，
   i. $(x^6+x^4+x^2+x+1)(x^7+x+1) \bmod m(x) = x^7+x^6+1$
e. x乘法运算：先乘x再模m(x)，相当于（在b7=0时）将b(x)表示的字节循环左移一位或（在b7=1时）先左移再将其与0x11B比特异或来实现

7. 〚$\boldsymbol{GF(2^8)}$〛$\boldsymbol{^4}$域上的多项式表示及运算
   a. 两个〚$GF(2^8)$〛^{4}域上的元素相加时，将两个元素对应多项式系数相加
   b. 两个〚$GF(2^8)$〛^{4}域上的元素相乘时，要将结果对一个特定的多项式取模
   c. 在AES加密系统中，〚$GF(2^8)$〛^{4}是在不可约多项式$\boldsymbol{M(x)=x^4+1}$上构造的有限域$<$〚$F(x)$〛$_{(M(x))},\oplus,\otimes>$
   d. 模M(x)的情况下，恰能保证a取{3,1,1,2}时有a逆取{b,d,9,e}
   e.

$$c(x)=a(x)\otimes b(x)=(c_6x^6+c_5x^5+c_4x^4+c_3x^3+c_2x^2+c_1x+c_0) \bmod (M(x))$$

这里，有 $c_6=a_3\cdot b_3$；$c_0=a_0\cdot b_0$；$c_1=a_1\cdot b_0\oplus a_0\cdot b_1$；$c_2=a_2\cdot b_0\oplus a_1\cdot b_1\oplus a_0\cdot b_2$；$c_3=a_3\cdot b_0\oplus a_2\cdot b_1\oplus a_1\cdot b_2\oplus a_0\cdot b_3$；$c_4=a_3\cdot b_1\oplus a_2\cdot b_2\oplus a_1\cdot b_3$；$c_5=a_3\cdot b_2\oplus a_2\cdot b_3$。

由于 $x^j \bmod (x^4+1)=x^{j \bmod 4}$，上式化简为 $d(x)=a(x)\otimes b(x)=d_3x^3+d_2x^2+d_1x+d_0$，其中

$d_0=a_0\cdot b_0\oplus a_3\cdot b_1\oplus a_2\cdot b_2\oplus a_1\cdot b_3$；$d_1=a_1\cdot b_0\oplus a_0\cdot b_1\oplus a_3\cdot b_2\oplus a_2\cdot b_3$

$d_2=a_2\cdot b_0\oplus a_1\cdot b_1\oplus a_0\cdot b_2\oplus a_3\cdot b_3$；$d_3=a_3\cdot b_0\oplus a_2\cdot b_1\oplus a_1\cdot b_2\oplus a_0\cdot b_3$

这个变换用矩阵表示为

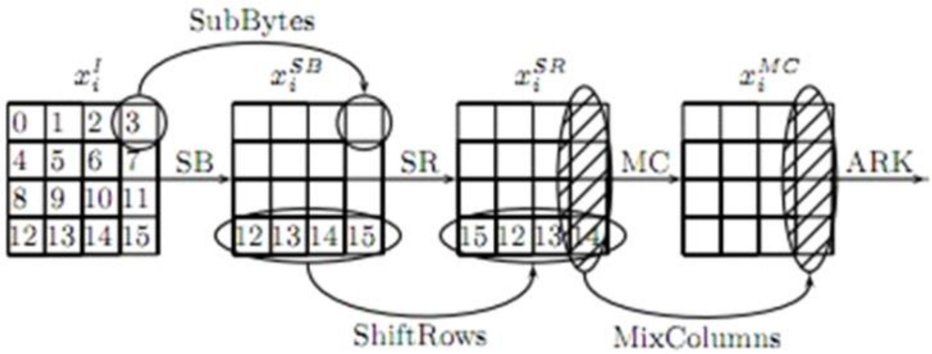
$$a(x)\otimes b(x)=\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix}=\begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix}\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- ◆
- ◆ AES

1. AES简介
   a. 明文长度和密文长度：128比特
   b. 密钥长度：128比特/192比特/256比特
   c. 轮数$\boldsymbol{N_r}$：10轮/12轮/14轮
   d. 分组长度N_b：4/6/8
   e. SPN结构代换置换网络substitution-permutation network
   f. 加密过程：
      i. 128bit明文P看成是16个$GF(2^8)$上的元素，
      ii. $X_0=P\oplus K_0$；
      iii. For i=1 to N_r-1

$$X_i=AK\circ MC\circ SR\circ SB(X_{(i-1)})$$

iv. 密文$C=AK\circ SR\circ SB(X_(N_r-1))$

g. 

h.

```
Cipher (byte in[4 * Nb], byte out[4 * Nb], word w[Nb * (Nr + 1)] )
{                                  // in、out 为明文分组输入和密文分组输出数组,w 为轮密钥数组
    byte State[4, Nb];                 // 定义一个状态矩阵 4 × Nb
    State = in;                        // 装入明文输入矩阵到状态矩阵
    AddRoundKey (State, w[0] );        // 明文信息和密钥混合,使用 w[0]开始的 Nb 个密钥
    for( int r = 1; r < Nr; r++ )      // 实现 1 到 Nr - 1 轮加密变换
    {
       SubBytes (State);
       ShiftRows (State);
       MixColumns (State);
       AddRoundKey (State, w[r * Nb]); // 使用从 w[r * Nb]开始的 Nb 个密钥
    }
    SubBytes (State);                  // 从这里开始最后一轮加密变换
    ShiftRows (State);
    AddRoundKey (State, w[Nr * Nb]);   // 结束轮加密变换,使用 w[Nr * Nb]开始的 Nb 个密钥
    Out = State;                       // 将最终的变化结果 State 矩阵,放到密文 out 矩阵中
}                                      // 结束加密变换,得到密文 out 矩阵
```

2. SB（对各字节做S盒非线性代换）

有限域 $GF(2^8)$上的仿射变换也对字节进行操作，设输入字节为$\{b_7b_6b_5b_4b_3b_2b_1b_0\}$，经过仿射变换后的输出字节为$\{b_7'b_6'b_5'b_4'b_3'b_2'b_1'b_0'\}$，则用矩阵来表示仿射变换表达式为：

a.

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

b. 从00到ff的输出如下
={0x63,0x7c,0x77,0x7b,0xf2,0x6b,0x6f,0xc5,0x30,0x01,0x67,0x2b,0xfe,0xd7,0xab,0x76,0xca,0x82,0xc9,0x7d,0xfa,0x59,0x47,0xf0,0xad,0xd4,0xa2,0xaf,0x9c,0xa4,0x72,0xc0,0xb7,0xfd,0x93,0x26,0x36,0x3f,0xf7,0xcc,0x34,0xa5,0xe5,0xf1,0x71,0xd8,0x31,0x15,0x04,0xc7,0x23,0xc3,0x18,0x96,0x05,0x9a,0x07,0x12,0x80,0xe2,0xeb,0x27,0xb2,0x75,0x09,0x83,0x2c,0x1a,0x1b,0x6e,0x5a,0xa0,0x52,0x3b,0xd6,0xb3,0x29,0xe3,0x2f,0x84,0x53,0xd1,0x00,0xed,0x20,0xfc,0xb1,0x5b,0x6a,0xcb,0xbe,0x39,0x4a,0x4c,0x58,0xcf,0xd0,0xef,0xaa,0xfb,0x43,0x4d,0x33,0x85,0x45,0xf9,0x02,0x7f,0x50,0x3c,0x9f,0xa8,0x51,0xa3,0x40,0x8f,0x92,0x9d,0x38,0xf5,0xbc,0xb6,0xda,0x21,0x10,0xff,0xf3,0xd2,0xcd,0x0c,0x13,0xec,0x5f,0x97,0x44,0x17,0xc4,0xa7,0x7e,0x3d,0x64,0x5d,0x19,0x73,0x60,0x81,0x4f,0xdc,0x22,0x2a,0x90,0x88,0x46,0xee,0xb8,0x14,0xde,0x5e,0x0b,0xdb,0xe0,0x32,0x3a,0x0a,0x49,0x06,0x24,0x5c,0xc2,0xd3,0xac,0x62,0x91,0x95,0xe4,0x79,0xe7,0xc8,0x37,0x6d,0x8d,0xd5,0x4e,0xa9,0x6c,0x56,0xf4,0xea,0x65,0x7a,0xae,0x08,0xba,0x78,0x25,0x2e,0x1c,0xa6,0xb4,0xc6,0xe8,0xdd,0x74,0x1f,0x4b,0xbd,0x8b,0x8a,0x70,0x3e,0xb5,0x66,0x48,0x03,0xf6,0x0e,0x61,0x35,0x57,0xb9,0x86,0xc1,0x1d,0x9e,0xe1,0xf8,0x98,0x11,0x69,0xd9,0x8e,0x94,0x9b,0x1e,0x87,0xe9,0xce,0x55,0x28,0xdf,0x8c,0xa1,0x89,0x0d,0xbf,0xe6,0x42,0x68,0x41,0x99,0x2d,0x0f,0xb0,0x54,0xbb,0x16}

c. 逆S盒
={0x52,0x9,0x6a,0xd5,0x30,0x36,0xa5,0x38,0xbf,0x40,0xa3,0x9e,0x81,0xf3,0xd7,0xfb,0x7c,0xe3,0x39,0x82,0x9b,0x2f,0xff,0x87,0x34,0x8e,0x43,0x44,0xc4,0xde,0xe9,0xcb,0x54,0x7b,0x94,0x32,0xa6,0xc2,0x23,0x3d,0xee,0x4c,0x95,0xb,0x42,0xfa,0xc3,0x4e,0x8,0x2e,0xa1,0x66,0x28,0xd9,0x24,0xb2,0x76,0x5b,0xa2,0x49,

0x6d,0x8b,0xd1,0x25,0x72,0xf8,0xf6,0x64,0x86,0x68,0x98,0x16,0xd4,0xa4,0x5c,0xcc,0x5d,0x65,0xb6,0x92
,0x6c,0x70,0x48,0x50,0xfd,0xed,0xb9,0xda,0x5e,0x15,0x46,0x57,0xa7,0x8d,0x9d,0x84,0x90,0xd8,0xab,0x0
,0x8c,0xbc,0xd3,0xa,0xf7,0xe4,0x58,0x5,0xb8,0xb3,0x45,0x6,0xd0,0x2c,0x1e,0x8f,0xca,0x3f,0xf,0x2,0xc1,0
xaf,0xbd,0x3,0x1,0x13,0x8a,0x6b,0x3a,0x91,0x11,0x41,0x4f,0x67,0xdc,0xea,0x97,0xf2,0xcf,0xce,0xf0,0xb4,
0xe6,0x73,0x96,0xac,0x74,0x22,0xe7,0xad,0x35,0x85,0xe2,0xf9,0x37,0xe8,0x1c,0x75,0xdf,0x6e,0x47,0xf1,
0x1a,0x71,0x1d,0x29,0xc5,0x89,0x6f,0xb7,0x62,0xe,0xaa,0x18,0xbe,0x1b,0xfc,0x56,0x3e,0x4b,0xc6,0xd2,0
x79,0x20,0x9a,0xdb,0xc0,0xfe,0x78,0xcd,0x5a,0xf4,0x1f,0xdd,0xa8,0x33,0x88,0x7,0xc7,0x31,0xb1,0x12,0x
10,0x59,0x27,0x80,0xec,0x5f,0x60,0x51,0x7f,0xa9,0x19,0xb5,0x4a,0xd,0x2d,0xe5,0x7a,0x9f,0x93,0xc9,0x9
c,0xef,0xa0,0xe0,0x3b,0x4d,0xae,0x2a,0xf5,0xb0,0xc8,0xeb,0xbb,0x3c,0x83,0x53,0x99,0x61,0x17,0x2b,0x4
,0x7e,0xba,0x77,0xd6,0x26,0xe1,0x69,0x14,0x63,0x55,0x21,0xc,0x7d}

3. SR（行移位线性变换）
    a.

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,0}$
$S_{2,2}$	$S_{2,3}$	$S_{2,0}$	$S_{2,1}$
$S_{3,3}$	$S_{3,0}$	$S_{3,1}$	$S_{3,2}$

    b. 密钥长度256时S2需循环左移3字节，S3不动
    c. 设运算需要循环左移i字节，则逆运算是循环左移Nb-i字节
4. MC（列混合线性变换）
    a. 详见GF(2^8)^4域的乘法运算，可用如下形式计算乘以M(x)
    b.

$$\begin{bmatrix} s'_{0i} \\ s'_{1i} \\ s'_{2i} \\ s'_{3i} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0i} \\ s_{1i} \\ s_{2i} \\ s_{3i} \end{bmatrix}$$

5. ARK（轮密钥加）及密钥扩展
    a.

首先，将 128 位的初始密钥写入密钥矩阵中：

k_0	k_4	k_8	k_{12}
k_1	k_5	k_9	k_{13}
k_2	k_6	k_{10}	k_{14}
k_3	k_7	k_{11}	k_{15}

那么，$w_0 = k_0k_1k_2k_3$，$w_1 = k_4k_5k_6k_7$，$w_2 = k_8k_9k_{10}k_{11}$，$w_3 = k_{12}k_{13}k_{14}k_{15}$。之后的每个轮密钥 w_i 要根据 w_{i-1} 和 w_{i-4} 来计算，如图 3-17 所示。

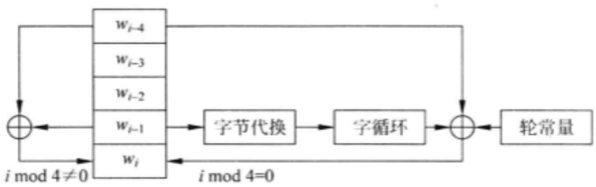


图 3-17 密钥扩展

当 i 不是 4 的倍数时，w_i 为 w_{i-4} 和 w_{i-1} 的异或。

当 i 是 4 的倍数时，则采用更复杂的计算方法：首先对 w_{i-1} 进行字循环，即将其 4 个字节循环左移一个字节。然后对结果进行字节代换，即根据 S 盒对 w_{i-1} 的每个字节进行字节转换。最后再与轮常量 RC 进行异或运算。轮常量也是一个 32 位字，但其右边 3 个字节总为 0。计算每个轮密钥用到的轮常量也不相同，当分组长度和密钥长度都为 128 位时，所用到的 10 个轮常量（最左边的一个字节）如表 3-6 所示：

表 3-6 轮常量

j	1	2	3	4	5	6	7	8	9	10
RC_j	01	02	04	08	10	20	40	80	1B	36

在轮密钥加变换中，轮密钥的各字节与状态中的各对应字节分别异或，实现状态和密钥的混合。

6. 密钥扩展伪码

```
KeyExpansion ( byte key[4 * Nk]), word w[Nb * (Nr + 1)], Nk)// 用来产生轮密钥 w 数组,w 以字为单位
{                                        // 原始密码 key 数组以字节为单位输入到函数中
    word temp;                           // 定义一个用来存放临时字变量的字
    i = 0;
    for ( i = 0; i < Nk; i++ )           // 原始密码 Key 数组放到前 w[0] - w[Nk - 1],用来密钥扩展
        w[i] = Word (key[4 * i], key[4 * i + 1], key[4 * i + 2], key[4 * i + 3]);
    for ( i = Nk; i < Nb * (Nr + 1); i++ )

    {                                    //扩展密钥,分 Nk = 4 和 6 或 8 两种情况
        temp = w[i - 1];
        if (i % Nk == 0)                 // % 表示整数的取模运算
          temp = SubWord ( RotWord ( temp ) ) ^ Rcon[ i / Nk];  // ^ 表示两个数的按位异或
        else if ( Nk == 8 && ( i % Nk == 4) )                  // && 表示两个数的逻辑与运算
            temp = SubWord ( temp );
        w[i] = w[i - Nk] ^ temp;         // 每次都要进行异或运算
    }
}
```

a.

b.

上面的函数 Word 用来将一个字的 4 字节,按由高到低表示成一个字(高位在前)。RotWord 函数将输入的一个字 $a_0a_1a_2a_3$(4 字节),循环左移一个字节后,重新组成一个字 $a_1a_2a_3a_0$ 输出;SubWord 函数对输入的字进行 SubBytes()变换(S 盒替换)后返回变换后的字。字数组 Rcon[i]由下面的方法得到:Rcon[i]=word (RC[i], {00}, {00}, {00}),其中 RC[i]为 x^{i-1} 在 $GF(2^8)$域上所代表的字节数值。如 RC[1]=01(为 x^0 代表的字节数),RC[i]={02}·RC[$i-1$],这里"·"为在 $GF(2^8)$域上的乘法。若 $i=36$, $N_k=4$,则 $i/N_k=9$,计算 RC[9]为 $x^8\bmod(m(x))=x^8\bmod(x^8+x^4+x^3+x+1)=x^4+x^3+x+1$ 所代表的字节{1b},即有 RC[8]={1b}。

c.

存在两个特殊情况:

(1) 对于在 N_k 整数倍处的密钥,在异或之前还将对这些字进行相应的变换,具体见上面的实现过程。

(2) 密钥长度为 256 位($N_k=8$)时的密钥扩展方案与密钥长度分别为 128 位($N_k=4$)及 192 比特($N_k=6$)时的密钥扩展方案稍有不同。当 $N_k=8$ 时,如果 $i-4$ 是 N_k 的整数倍,则在进行异或之前,需要先对进行 Subword()变换。

7. C语言实现要点
   a. 二维矩阵对应数组指针int (*p)[m]；p++偏移sizeof(int)*m
   b. 二级指针对应指针数组int *p[m]；p++偏移sizeof(int *)
   c. *p++先取后加；(*p)++先取后给值加；*++p先加后取；++*p先取后给值加
   d. 注意数组名本身是常量，取地址也改不了，应让另一个指针=数组名